In the Claims:

1-22 (cancelled)

23. (new) A method of a software architecture that provides high versatility and performance, the method comprising the steps of: Having two dimensions, an application dimension and a core dimension; Relating the application dimension to the different applications; Relating the core dimension to the applications dimension; Having a CORE, where said CORE is a complex organizational referee engine which manages information and interaction between different extensions of the architecture attached to it and is a complex compound of software components that can accept different kinds of software extensions dynamically in a hot plug fashion, and exchange information with other instances of itself, within the core dimension related; Having the CORE within the core dimension share information and integrate the system architecture; Having applications based on abstractions that are composed of drivers, abstraction layers and a unique CORE; Having a plurality of terminal devices, having said CORE consists of a complex compound of software components that dynamically accepts software extensions and exchanges information with other COREs where said abstraction layers consists of a software layer the hides implementation detail and data structures of a specific software and said module extensions are comprised of an abstraction layer and a driver.

24. (new) The method of claim 23 in which said core dimension consists of a plurality of COREs connected by a bidirectional communication means.

25. (new) The method of claim 23 in which said CORE consists of three parts; i) a CORE Engine kernel, which manages the task processing and scheduling, manages the information exchange between the CORE dimension, and manages the dynamic extensions, ii) an InterExtension Communication and logic manager which manages the

CORE's communication and tasks with a plurality of extensions, iii)an InterCORE Communication manager which manages the CORE's communication with a plurality of other COREs.

26. (new) The method of claim 23 in which said abstraction layer consists of two parts; i)a CORE-Abstraction interface, which interfaces an extension with a CORE; and ii)Extension Knowledge Layer, which contains logic and knowledge about the operations of extensions.

27. (new) The method of claim 23 in which includes the step of having an extension driver layer that consists of two parts; i)an Abstraction-Driver interface, which interfaces the Abstraction layer with a terminal device; and ii)Driver logic used to control the terminal device.

28. (new) The method of claim 23 used for a Control and Automation Application.

29. (new) The method of claim 23 used for an Assets Control application.

30. (new) A method to replace a terminal device with a new terminal device using the method in claim 23 consisting of the adding the step of changing the driver for the extension.

31. (new) A method to add a new terminal device to a system using the method in claim 23 consisting of adding the steps of: a)constructing a new extension for the terminal device; b)interfacing the new extension into the CORE; c)asking the CORE for the

required data and information to handle the new extension.